ITL.1091US
(P18429)

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: UPDATING CODE WITH VALIDATION

INVENTOR: ROBERT HASBUN

Express Mail No. EV 337934534 US

Date: March 30, 2004

Prepared by: Trop, Pruner & Hu, P.C., Mark J. Rozman
8554 Katy Freeway, Ste. 100, Houston, TX 77024
713/468-8880 [Office], 713/468-8883 [Fax]

UPDATING CODE WITH VALIDATION

Background

Typically updates for wireless devices, such as cellular telephone handsets, are done in a factory or at an authorized service center because of the risk of downloading corrupted or malicious code, which could impact handset functionality and network provider revenue. As handsets move from a closed architecture environment to rich media and services, a method to enable new features, remedy bugs, update protocol code, and the like, in wireless devices is desirable.

Typically, such updates occur by receiving data packets at the communications portion of the device and providing the packets to an applications portion of the device for manipulation into a file, and then sending the file back to the communications portion. Because remote updates not done in a factory or service center are exposed to viruses, hacking and other malicious actions, methods to verify that a file came from a trusted source and that the file is not corrupted exist for the applications portion. However, updating the communications portion of the device is problematic. The communications subsystem does not have the processing power, code or data space to download and verify a full update in an autonomous fashion. Further, it would not be practical for the communication subsystem to include flash management, file-system support, cryptographic and other security capabilities, as multiple such code portions would exist in the wireless device, potentially causing conflicts, compatibility, resource, and implementation issues. Thus a need exists for a communications subsystem to verify that an update is trusted autonomously from the applications subsystem. Further, to enable updates, upgrade a device, remedy bugs, or to upgrade a communications portion of a device outside of the service center or factory (e.g., over-the-air (OTA)), a non-spoofable method of verifying the trust state of the applications subsystem is required.

Brief Description of the Drawings

FIG. 1 is a flow diagram of a method in accordance with one embodiment of the present invention.

FIG. 2 is a block diagram of a register in accordance with one embodiment of the present invention.

FIG. 3 is a block diagram of a system in accordance with one embodiment of the present invention.

Detailed Description

Updates of code, data, and other information, and secure operations intended for a communications portion of a wireless device may be performed in accordance with various embodiments of the present invention. As used herein, the term "information" may refer to both data and/or instructions. While the device may vary, in certain embodiments, a wireless device may be a cellular telephone (e.g., of a 2.5 generation (G) or later variety), a personal digital assistant (PDA), or a notebook computer, for example.

Such updates can vary in different embodiments. In certain embodiments, the updates may relate to communications or software updates to remedy bugs. Alternately, updates may include downloading of any other desired digital content into the wireless device. While such content may take various forms, in certain embodiments the digital content may be secure content, such as rights management data or keys to manage video or audio file access and the like. Further, in certain embodiments, digital content may be applications desired by a user of the wireless device. The initiation of downloading updates to the wireless device may vary: for example, updates to code may be initiated by a service provider, while downloading of multimedia digital content may be initiated by the user.

Referring now to FIG. 1, shown is a flow diagram of a method in accordance with one embodiment of the present invention. As shown in FIG. 1, method 10 may begin by receiving a code update (block 20). While shown in the embodiment of FIG. 1 as receiving a code update, it is to be understood that in different embodiments, various information types may be received.

In certain embodiments, upon receipt of such code, an update manager or loader (which may be in an applications subsystem) may verify (by public/private key and/or other robust methods) the credibility of the external update code, the integrity of the code to be updated, as well as ensuring that the code was sent by a trusted source, and meant for download to the device. For example, in certain embodiments, data packets received via an air-interface protocol such as tunneled transmission control protocol/Internet protocol (TCP/IP) packets, may be received by a communications processor and forwarded directly to the applications processor for decryption and resolving into a file.

After such data packets are resolved into a file and stored in memory (e.g., memory associated with the applications processor), the file may be validated using the same robust cryptographic code the applications processor would use to verify any other file (e.g., triple data encryption standard (3DES) or advanced encryption standard (AES) for encryption, Secure Hash Algorithm (SHA) for integrity checking, and Rivest Shamir Adelman (RSA) for signature verification, or the like) to ensure that the file has not been permuted and that the code is authenticated. In certain embodiments, various security processes, such as performing signature verification, hashing and other operations may occur, as well known to those of skill in the art. Upon validation of the code, it may be stored in a predetermined location in memory in which code update portions are stored prior to the update process. In one embodiment, the memory may be a nonvolatile memory.

After a successful download, the applications subsystem may initiate itself into a trusted state (block 30). For example, the applications subsystem may cause itself to be reset into a trusted state. In various embodiments, a trusted state may perform trusted processes from a trusted code base, which may be a minimal code base. However, normal application programs of the device, such as communications and data processing, may not be performed in the trusted state, but only after the trusted state is exited.

Such a trusted state may occur immediately or may be deferred until a later time. For example, the processor may vector to a fixed read-only memory (ROM) location. In certain embodiments, initiation into a trusted state may be performed by causing the wireless device to restart through a power-on self test (POST) operation. For example, such a POST routine may be performed upon booting up the wireless device (i.e., upon a hard reset). However, in other embodiments, a trusted state may be initiated without a power on reset of the device.

It is to be noted that during the reset and initiation of a trusted state, a hardware asset of the wireless device may be implemented to provide a non-spoofable indicator of the state of the applications subsystem (i.e., that it is in a trusted state). For example, in one embodiment, a one-way register (as will be discussed further below with regard to FIG. 2) may be cleared upon reset or entering of a trusted state. In other embodiments, another hardware asset, such as a voltage level generator, or private data signal or strobe may be used to provide such an indication of a trusted state.

In certain embodiments, upon reset, the applications processor may look for an updated code portion, either at a particular location or by a known indicator. Then the applications subsystem may determine whether the code update is verified (diamond 40). For example, in one embodiment, the applications processor may verify the cryptographic validity of the image as described above.

While discussed herein as including an applications subsystem and a communications subsystem, it is to be understood that in other embodiments, methods may be used in connection with a system having a transmitting subsystem (i.e., a sender of a code update or the like) and a receiving subsystem (i.e., a receiver of such an update or the like).

If the code update is not verified (e.g., the cryptographic or other verification method fails), the update may be terminated (block 45). Furthermore, the hardware asset, e.g., the one-way register, may be set by the applications subsystem to provide an indication that the applications subsystem is about to leave a trusted state (block 80). While shown in the embodiment of FIG. 1 as setting the register, in other embodiments, the method may be performed by resetting the register to indicate that the trusted environment is about to be exited. Of course in other embodiments, a different hardware asset may be manipulated to indicate that the system has left its trusted state.

Finally, the trusted state may be exited (block 85). At such time, remediation for the failure to update may be undertaken, such as a virus scan, verification of the platform software, or the like. If appropriate, normal system operations may commence, and one or more desired application programs may be executed. Because of the failure of verification of the update, no update is applied to the communications subsystem.

If instead the code update is verified at diamond 40, a receiving portion of the system may be informed of the availability of the update (block 50). In this instance, note that the one-way register is not set and accordingly remains in its initial (i.e., logic 0) state. The indication of update availability may be sent from the applications subsystem to the communications subsystem on a desired channel. For example, a data packet including such a message may be sent on a scalable link between the two subsystems.

When the communications subsystem receives the indication, it may check the one-way register to determine whether it is set (diamond 60). If the register is not set, this indicates that the applications subsystem is in a trusted state and that the indication of a code update and the content of the code update itself may be trusted.

Accordingly, the desired update may be downloaded to the communications subsystem (block 70). In other embodiments, rather than a code update, a software patch or new code download, or a trusted operation may be performed as instructed or delivered by the applications subsystem. In yet other embodiments, other types of content for a communications subsystem may be sent in accordance with such a method. Such content may include a validated driver or other updated software, a configuration file, or the like.

In various embodiments, the communications subsystem may copy the update code into its memory in any one of a number of well-known manners, and may use the code to program and run the communications subsystem after its receipt and storage.

When the update code or other action is successfully downloaded or communicated to the communications subsystem, the one-way register may be set as discussed above (block 80), and the applications subsystem trusted state may be exited (block 85), also discussed above.

If instead the communications subsystem determines at diamond 60 that the register is set, the update is terminated (block 90). Of course, as discussed above in other embodiments, a different hardware asset may be used to indicate the trust state of the application subsystem.

Furthermore, in certain embodiments in addition to terminating the update, the communications subsystem may perform a remediation event or send an indication to the applications subsystem to remediate if appropriate. For example, in certain embodiments the subsystem may cause all communications to be terminated and precluded, or the subsystem may set a nonvolatile indication for the applications subsystem that there was a failure to update due to lack of trust. Such an indication may be read by the applications subsystem when it is in its trusted state, in certain embodiments. Alternately, a communications subsystem may, upon receiving a request to update outside of a trusted state, send a message to a wireless service provider, or other central location to advise of the presence of a potential virus or other malicious code.

Referring now to FIG. 2, shown is a block diagram of a register in accordance with one embodiment of the present invention. As shown in FIG. 2, register 100 may be a D-type flip-flop register. While shown in the embodiment of FIG. 2 as a flip-flop, it is to be understood that in other embodiments, any register, memory location, or other hardware asset may be used to indicate a trusted state of an applications processor, boot process, and/or a code update.

While such register may be located in various places, in one embodiment, the register may be located such that the applications processor may cause the register to be set, while the communications processor may read the register's contents. In one embodiment, both the applications processor and the communications processor may cause the register to be set. In another embodiment, the register may be located in the communications processor itself.

As shown in FIG. 2, register 100 includes an input (V_{cc}) and a clock pulse (CLK) that may be an Applications Set signal received from an applications processor of the wireless device when leaving its trusted state. Also, a reset input may receive a Reset signal that may be received by register 100 upon hard reset of the wireless device, and thus may be used to reset register 100 on power up or other entry into a trusted state. In one embodiment, when the Applications Set signal is input into register 100, a high value may be output on a data output line (D_{out}). Such an output signal may be read by the communications processor to learn the state of the applications processor, boot state, and/or code update. While in the embodiment discussed, a programmed (i.e., set) or high signal may be indicative of an untrusted state (e.g., a post-trusted boot state) and a low or reset state indicative of a trusted state (e.g., a trusted boot state), embodiments of the present invention are not so limited.

In various embodiments, register 100 may be a one-way register that once set cannot be reset without resetting the system. That is, register 100 may be writable by the applications processor (changeable) once per hardware-reset period (i.e., one-way). In such manner, if a register value is a low or reset state, a communications processor may reliably determine that the applications processor is in a trusted state (e.g., a trusted boot). In other words, during a booting procedure of the device (i.e., a trusted boot), by definition the register may be reset during the hard reset and may be programmed/set by the applications processor before vectoring out of the boot code.

In such manner, the communications portion may avoid performing a security process, such as a public/private key handshake or other heavy process, thus unifying security processes to run on the applications processor only, and simplifying operations on the communications processor.

Embodiments may be implemented in a program. As such, these embodiments may be stored on a storage medium having stored thereon instructions which can be used to program a system to perform the embodiments. The storage medium may include, but is not limited to, any type of disk including floppy disks, optical disks, compact disk read-only memories (CD-ROMs), compact disk rewritables (CD-RWs), and magneto-optical disks, semiconductor devices such as read-only memories (ROMs), random access memories (RAMs), erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), flash memories, a phase change or ferroelectric memory, a silicon-oxide-nitride-oxide-silicon (SONOS) memory, magnetic or optical cards, or any type of media suitable for storing electronic instructions. Similarly, embodiments may be implemented as software modules executed by a programmable control device, such as a computer processor or a custom designed state machine.

Referring now to FIG. 3, shown is a block diagram of a wireless device with which embodiments of the invention may be used. While discussed in FIG. 3 as a wireless device, it is to be understood that other embodiments may be used with other types of systems, including desktop computers, servers, and the like. As shown in FIG. 3, in one embodiment wireless device 200 includes an applications processor 210, which may include a general-purpose or special-purpose processor such as a microprocessor, microcontroller, application specific integrated circuit (ASIC), a programmable gate array (PGA), and the like. Applications processor 210 may be used to execute various applications such as data processing functions, modification and manipulation of digital content and the like. In one embodiment, applications processor 210 may be a 32-bit processor, such as an XScale™ processor, available from Intel Corporation, Santa Clara, California.

Applications processor 210 may be coupled to a communications processor 220, which may be a digital signal processor (DSP) based on a micro signal architecture via an internal bus, which may include a scalable link 225 (such as a mobile scalable link), which may be formed of a plurality of gating devices to scalably transfer data between the processors. A memory subsystem 230 may be coupled to both applications processor 210 and communications processor 220, in certain embodiments. Memory subsystem 230 may include both volatile and non-volatile memory, such as static RAM (SRAM), dynamic RAM (DRAM), flash memories, and the like. While shown in FIG. 3 as separate components, it is to be understood that in other embodiments two or more of the components may be integrated into a single device, such as a single semiconductor device.

It is to be understood that communications processor 220 may include various functionalities including wireless communication with external sources. For example, communications processor 220 may include a wireless interface (which in turn may have an antenna which, in various embodiments, may be a dipole antenna, helical antenna, global system for mobile communication (GSM) or another such antenna). In certain embodiments, the wireless interface may support General Packet Radio Services (GPRS) or another data service. GPRS may be used by wireless devices such as cellular phones of a 2.5G or later configuration.

Other embodiments of the present invention may be implemented in a circuit switched network such as used by 2G technologies, a Personal Ccmmunications System (PCS) network, a Universal Wireless Telecommunications System (UMTS), or UMTS Telecommunications Radio Access (UTRA) network or other communication schemes, such as a BLUETOOTH^{TM} protocol or an infrared protocol (such as Infrared Data Association (IrDA)).

While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.